Temporal Scalable Visual SLAM using a Reduced Pose Graph

Temporally Scalable Visual SLAM using a Reduced Pose Graph

Hordur Johannsson, Michael Kaess, Maurice Fallon and John J. Leonard

***Abstract*—In this paper, we demonstrate a system for temporally scalable visual SLAM using a reduced pose graph representation. Unlike previous visual SLAM approaches that use keyframes, our approach continually uses new measurements to improve the map, yet achieves efficiency by avoiding adding redundant frames and not using marginalization to reduce the graph. To evaluate our approach, we present results using an online binocular visual SLAM system that uses place recognition for both robustness and multi-session operation. To allow large-scale indoor mapping, our system automatically handles elevator rides based on accelerometer data. We demonstrate long-term mapping in a large multi-floor building, using approximately nine hours of data collected over the course of six months. Our results illustrate the capability of our visual SLAM system to scale in size with the area of exploration instead of the time of exploration.**

I. Introduction

To achieve long-term robotic autonomy, in complex and dynamic environments and independent of duration, requires mapping algorithms which scale solely with the area explored and not in exploration time. There are many applications for autonomously navigating mobile robots, such as service, delivery, and search and rescue, in which long-term mapping and localization operations are critical. To be widely applicable, the system has to construct a map of its operating environment using only onboard sensors, and continuously update and extend this map with new information as the world changes.

Many recent solutions to mapping are based on the pose graph formulation [18]. In this formulation the world is represented by a set of discrete poses sampled along the full trajectory of the robot, which are connected by odometry and loop closure constraints. Very efficient recursive algorithms have been presented which can maintain an online solution to this continuously expanding optimization problem, however the pose graph, by design, grows unbounded in time as recognized by Biber and Duckett [1]. This is true even for small environments which are repeatedly explored, making the naive application of the pose graph unsuitable for long-term mapping.

It is desirable to achieve a persistent mapping solution that scales only in terms of the spatial extent of an environment, and not the duration of the mission. Additionally, long-term persistent operation will require the ability to develop compact representations, which can effectively describe an environment of interest, yet still provide robustness to changes in the environment and recovery from mistakes.

The primary contribution of this paper is an approach, called the reduced pose graph, that addresses the temporal scalability of traditional pose graphs. For long-term mapping, the size of the optimization problem should be bounded by the size of the explored environment and be independent of the operation time. To achieve this goal, the reduced pose graph reuses already existing poses in previously mapped areas, keeping the number of poses bounded by the size of the explored environment. A key insight is that new measurements can still be used to further improve the map, by converting them into constraints between existing poses. The process is fluid, with new poses being added when new spaces are being explored.

The advantages of the reduced pose graph extend beyond scalability. Our approach maintains multiple constraints between each pair of poses. While these multiple constraints could be combined immediately, retaining redundancy allows for constancy checking and the detection of faulty constraints. In combination with a robust estimator this can limit the effect of erroneous constraints on the state estimation. When consensus is reached over a significant number of constraints, they can eventually be combined into a single constraint, avoiding incorporating bad constraints into the combined edge.

Our secondary contribution is a full 6-DOF visual SLAM system which we use to illustrate and evaluate the proposed reduced pose graph formulation. Our system is stereo-vision-based and operates in real-time and has been tested with data from multiple robotic platforms. A visual odometry model produces incremental constraints between key-frames which are used as input to the reduced pose graph. A place recognition module uses appearance-based methods to propose loop closures for which a geometric consistency check provides the actual constraint if successful. Place recognition allows mapping over multiple sessions, and provides improved robustness in the case of localization failure.

Robustness can be further improved by using other sources of egomotion. In our work we have utilized wheel odometry and an IMU. An accelerometer is particularly useful to eliminate drift in inclination which can accumulate in explorations as large as those presented here.

To allow for operation in large multi-floor indoor environments, our SLAM system can automatically detect elevator transitions. Using an accelerometer, this approach detects characteristic elevator motion to track the vertical displacement of the robot.

We have evaluated our approach on data recorded with a PR2 mobile robot from Willow Garage. We use several hours of data corresponding to eleven kilometers of robot trajectory recorded over a period of several months, demonstrating robustness to changes in the environment. Our system has

been tested with both stereo and RGB-D data from various robot platforms. We include a model generated from Kinect data for illustration purposes.

II. Related work

The pose graph optimization approach to SLAM was first introduced by Lu and Milios [18] and further developed by many researchers including Gutmann and Konolige [11], Folkesson and Christensen [8], Dellaert [5] and Olson *et al.* [23]. Significant research has focused on providing efficient solutions, both approximate and exact. Notable examples include hierarchical representations [10], collections of local maps [22, 7, 2] as well as relative and non-Euclidean) approach [21, 26]. However few have addressed reducing the growth in size of the number of pose graph nodes as a function of time.

The visual maps by Konolige and Bowman [16] are closely related to our work. They create a skeleton graph of views similar to a pose graph. To keep the density of views or poses constant in a given region, least-recently used views are removed from the skeleton by marginalization. Our work, in contrast, avoids marginalization by not adding redundant views to begin with. Other related recent work in multi-session visual SLAM was presented by McDonald *et al.* [19], which combines multiple mapping sessions in a pose graph optimization framework, with appearance-based loop closing [3], however this work did not address temporal scalability.

Compact pose SLAM by Ila et al. [13] uses an information-theoretic method to decide which constraints should be added. New poses are only added to the estimator (an information filter) if no other poses are nearby, while taking into account information gain from potential loop closures. The paper does not address how to limit growth when continuously operating in the same environment. In contrast, our approach can connect constraints to existing poses in areas already mapped — so as to avoid the need for the periodic addition of new nodes along the trajectory.

Kretzschmar et al. [17] also use an information-theoretic approach to decide which laser scan should be removed from the graph. They have shown large reductions in complexity for laser-based pose graphs, using an approximate marginalization to retain the sparsity of the solution.

Also related to this are the sample-based maps by Biber and Duckett [1]. An initial map is created with traditional SLAM methods, and then updated at multiple different time scales to capture dynamic changes in the environment. The map is represented by a set of evolving grid-based local maps connected to an underlying pose graph. They demonstrate long-term mapping on several hours of data recorded over five weeks. This work is specific to laser-range data.

For monocular SLAM, a different approach without pose graphs has been taken for managing complexity when repeatedly mapping the same environment. Most notably, Klein and Murray [15] introduced monocular parallel tracking and mapping (PTAM), where a map of sparse features is updated over time by bundle adjustment, and the camera is continuously localized based on this map. Targeting augmented reality applications, PTAM is limited to small scale environments, mostly because of the complexity of bundle adjustment.

An extension to augmented reality applications to large-scale environments using several local PTAM maps is demonstrated by Castle et al. [4]. More recently, Pirker et al. [25] proposed larger scale monocular reconstruction again based on bundle adjustment. Their system updates the map in dynamic environments and achieves real-time performance with the exception of loop closures.

Eade et al. [6] reduces complexity by marginalization and degree thresholding for monocular SLAM with odometry. When the degree of a node exceeds a specific threshold, the constraint with the least residual error is removed. While suitable for low-power platforms, the estimate will be biased towards measurements with larger errors.

A completely different approach to long-term mapping is taken by Milford and Wyeth [20] in biologically inspired work. Their previous RatSLAM system used panoramic video and odometry as perceptual stimulus. While their work does not try to produce Cartesian maps, they have demonstrated impressive long-term mapping and navigation in dynamic environments.

III. Reduced pose graphs

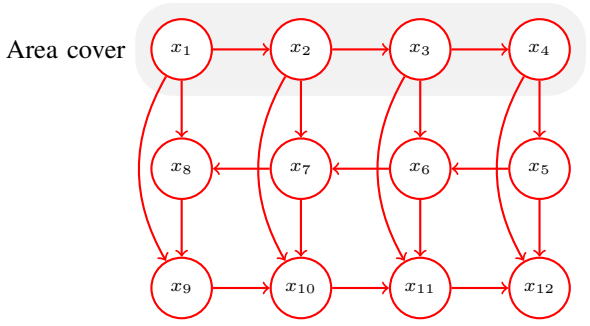
A pose graph exploits the fact that a map of the environment can be generated from a set of localized robot poses and their sensor measurements. Select poses along the robot trajectory are represented by nodes in the pose graph, typically created at fixed time intervals or after moving for a certain distance.

Independent on the application domain, constraints between consecutive poses are added based on incremental odometry (laser, wheel, visual or sonar) while loop closing constraints are found between two arbitrary poses (based on laser-scan, feature or visual matching). For this set of constraints, optimization finds the best configuration for all poses and is often formulated as a maximum likelihood problem. An explicit map can be formed by projecting the sensor measurements at each pose into a common reference frame. By construction, this basic pose graph formulation has a significant drawback: the graph will grow unbounded with time as new poses are constantly being added.

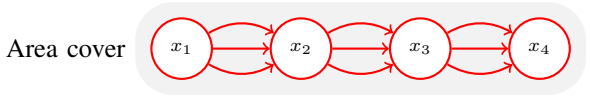
However for long-term operation in a fixed size environment, say a large building, a bounded computational cost is a key requirement. An ideal solution fulfills the following requirements:

- the optimization should remain efficient; and
- the map can be corrected if an error is detected.

The motivation for the reduced pose graph is to reuse existing poses in previously mapped areas. The concept of pose graph reduction to achieve temporal scalability is intuitively appealing, and indeed has been proposed in the previous literature in several contexts [9, 28, 29]. Pose graph reduction is related to the use of keyframes in PTAM Klein and Murray [15], but the use of keyframes alone presents several shortcomings which we seek to avoid in our approach. In particular,



(a) Traditional pose graph



(b) Our reduced pose graph

Fig. 1. Comparing our reduced pose graph with the traditional solution on a small example. The same environment is traversed three times. The area is covered by the first four poses. The reduced pose graph reuses these poses and transform new constraints into constraints between the existing poses.

using new information only to track the robot/camera pose results in substantial loss of information vs. our approach which continues to make use of new measurement information to improve the map. This strategy corresponds to the approach taken by the exactly sparse extended information filter (ESEIF) [28, 29] to maintain sparseness and preserve consistency in an Information Filter context. A related similar strategy has been previously adopted in a pose graph context by Grisetti *et al.* [9]. The challenges in implementing such an approach include deciding when to add new poses, how to minimize the loss of information, and how to achieve robust performance for long-term operation.

Our technique for pose graph reduction is illustrated in a small example in Fig. 1. The figure shows a traditional pose graph at the top, and a reduced pose graph at the bottom. A small environment is traversed three times, with four pose nodes added for each traversal. Loop closure constraints connect to previously created poses at the same location. The reduced pose graph at the bottom of the figure is much smaller, with only four nodes. After the first traversal, existing nodes are reused, and the new measurements are converted into constraints between the original four poses.

In practice, exactly the same pose is never revisited, as a result we shall define an area for which a specific pose is valid and representative. Essentially this means partitioning the explored space. How the world is partitioned will depend on the sensors used and the environment the robot operates in. In this work the primary sensor is a forward looking stereo camera. Our partition uses a regular grid over 2D position and heading. When the map estimate is updated, the location of poses might change, causing more than one pose to fall in the same partition. In that case the most recent one is retained as the active one.

The reduced pose graph consists of pose nodes $X = \{x_i\}$ and constraints $Z = \{z_k\}$. A pose node contains the actual pose as well as the sensor measurement needed to construct a map and to recognize when the pose is revisited. A constraint z_k measures the spatial relationship of two poses i_k and j_k. The configuration of poses that best fits all measurements is given by the the maximum likelihood solution

$$X^* = \underset{X}{\operatorname{argmin}} \sum_k C_k(f(x_{i_k}, x_{j_k}) - z_k)$$

where f is a function that predicts a constraint from two poses and C_k is a cost function associated with constraint k. The same solution applies to the traditional pose graph, with the only difference being the way that constraints and nodes are created.

Let us consider how nodes and edges are added to graph. The system starts by creating a pose graph consisting of the first pose alone. By default this pose is defined to be the active pose and its is used as the reference. We continuously track our uncertain transformation (z_0, Σ_0) from this currently active node using (visual) odometry.

During exploration, the graph can change under one of the following three conditions:

1) Robot leaves range of active pose: a new node, x_1, is created and a constraint $|f(x_0, x_1) - z_0|_{\Sigma_0}$ is added to the graph. The new node x_1 is initialized as $x_0 \oplus z_0$.
2) Loop closure to a pose different from the active one: the loop closure yields a transformation z_1 from the current pose to another node x_1. A new constraint $z_0 \oplus z_1$ with covariance Σ_{01} (see below) is added to the graph. The active pose is now updated to x_1.
3) Loop closure to the active pose: if the new registration has higher entropy, i.e. $det(\Sigma_1) > det(\Sigma_0)$ then this registration is set as the transformation to the active location.

Compounding of uncertain transformations follows Smith et al. [27]. Given a chain of transformations z_{12}, z_{23} with covariances Σ_{12} and $\Sigma 23$ respectively, the compounded transformation z_{13} is computed as follows:

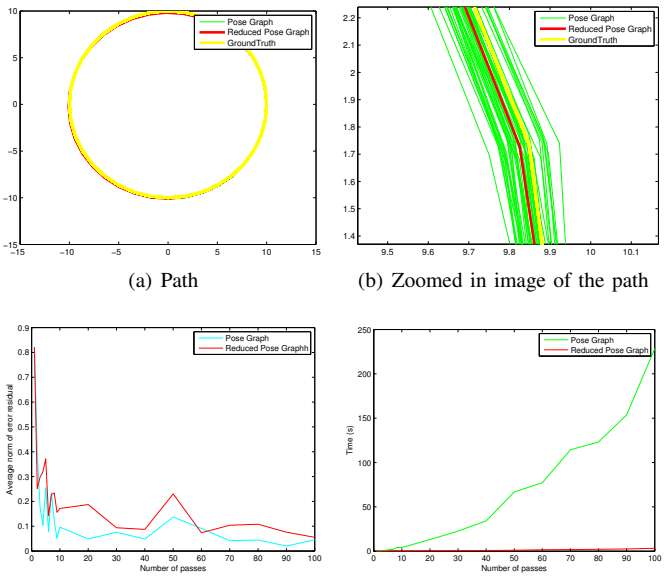
$$z_{13} = z_{12} \oplus z_{23}$$
$$\Sigma_{13} = = J_{1\oplus}\Sigma_{12}J_{1\oplus}^T + J_{2\oplus}\Sigma_{23}J_{2\oplus}^T$$

This is the first order approximation of the mean and covariances, where z_{12} and z_{23} are assumed to be independent. Now the factor added to graph will be $|f(x_1, x_3) - z_{13}|_{\Sigma_{13}}$. Where f is a function that computes the transformation between x_1 and x_3, i.e. $x_3 = x_1 \oplus f(x_1, x_3)$.

When a loop closure constraint is added the measurement z_1 that closed the loop should not be reused, to prevent overconfidence in the estimate. Instead, a new loop closure should be acquired to start a new chain of measurements. Alternatively, if enough inliers were found in the loop closure, they could be split into two sets, one for the loop closure and the other for re-localizing. In comparison a full pose graph approach would have added a new pose at the point of loop closure. While some odometry information is discarded in this manner,

we have seen minimal consequences for this approximation when performing long-term mapping operations, with repeated traversals of the environment. A comparison of a full pose graph vs. the reduced pose graph is shown in Fig. 2. It shows that this reduction achieves similar accuracy while greatly reducing the computation time.

One of the benefits of this representation is that it reduces the number of variables in the graph. In addition most of the updates are the addition of new constraints (but not variables) to the estimation problem, which can be applied incrementally in an efficient manner using the iSAM algorithm [14] for pose graph optimization.



(a) Path (b) Zoomed in image of the path

(c) Absolute error (d) Timing

Fig. 2. This figure shows the results of state estimation for a simulated vehicle traversing along a circular trajectory and visiting the same places in each pass. The results are shown for a full pose graph and a reduced pose graph. (a) and (b) show the estimated trajectories for 100 passes. (c) shows the mean error for each method and (d) shows the timing.

IV. Visual SLAM

In this section we apply the reduced pose graph concept to visual SLAM. An architecture overview of the system is given in Fig. 3. The main components are: visual odometry, loop proposal, loop closure, map management, and the map estimation. In typical operation the visual odometry algorithm is run on a single thread at 30Hz (the camera framerate) while the remaining components are evaluated on a second thread.

a) Visual Odometry: Incremental motion is estimated using an efficient implementation of stereo-visual odometry. This approach, named FOVIS (Fast Odometry for VISion) is explained and quantified in more detail in Huang et al. [12].

In brief, FAST features are extracted approximately uniformly across a smoothed gray-scale version of the image. An initial estimate of rotation is made. Using this estimate, feature-to-feature matching is carried out and refined to sub-pixel accuracy. The distance between pairs of features in

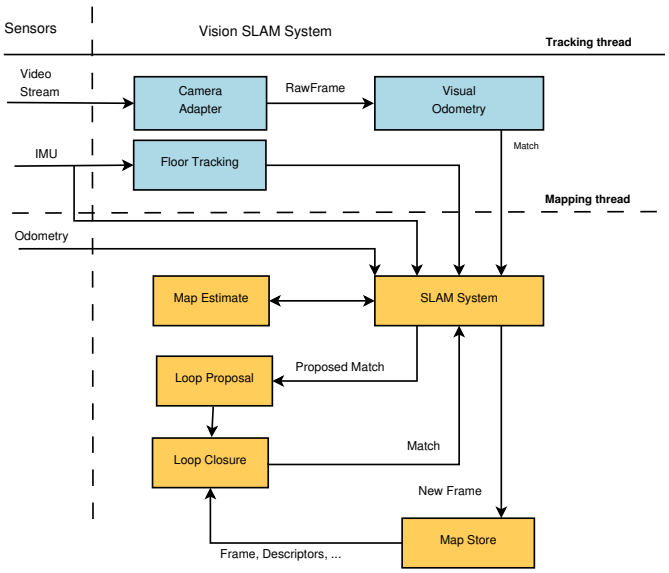


Fig. 3. Architecture for the proposed visual SLAM system. On the left are the sensor inputs and on the right are the clients that consume information from the visual SLAM system. The FOVIS library is used for visual odometry [12] and geometric verification of loop closures. Dynamic Bag of Words [24] and the vocabulary tree in ROS have been used for loop proposals. The OpenCV library is used for computing feature descriptors and matching. The map estimation uses the iSAM library.

consecutive frames is compared for constancy. The largest set of mutually corresponding feature pairs are declared to be the inlier features. Relative motion from frame to frame is then estimated as the transformation minimizing the re-projection error of the inlier features. This estimate is then passed to the map management module where it is combined with the last position estimate to give the current position estimate.

This implementation supports both stereo cameras and RGB-D sensors (such as the Microsoft Kinect) and results for both camera types are presented in Section V.

Additionally, our approach can incorporate IMU (roll and pitch) and wheel odometry (horizontal translation) in situations in which the vision system cannot estimate motion, e.g. featureless walls, low light, and occlusion of the camera. If required, it is also possible to use the wheel odometry as the sole source of relative motion estimation — using vision data only for the detection of loop closures.

b) Map Management: The map management module receives incoming measurements from the visual odometry, IMU and other sensor inputs. For any incoming frame a feature descriptor is computed for each keypoint in the new frame. The feature descriptors are maintained for later use in the appearance-based loop proposal and frame registration modules. Several different descriptor types are supported by our implementation including BRIEF, Calonder and SURF. In each case we utilized the OpenCV implementations.

Another responsibility of the map management module is to determine reasonable features to consider for loop closures. A place map is maintained by partitioning the explored space using a regular grid in 3 dimensions (x, y, and heading). If

multiple nodes are assigned to the same grid cell, the node most recently added is selected as the active node. This map is then used to actively localize the robot by continuously registering the current image frame with this active node.

c) Active Node Registration: To register two frames to one another a collection of putative matches are found by matching each keypoint to the keypoint that has the closest feature descriptor. This is done using brute-force matching. The descriptor we most commonly use is the BRIEF descriptor, which can be compared very efficiently. Using the feature depth estimates, the same clique-based method mentioned previously for visual odometry is retained. Finally the 6-DOF transformation between the two frames is estimated by minimizing the re-projection errors of these matched keypoints. If the number of inliers is within a threshold the registration is accepted.

d) Global Node Registration: However, if registration to the active node fails, a global loop closure algorithm (across the entire set of poses) is used instead. A bag of visual words is used to describe each frame and using an inverted index an efficient search is carried out for similar frames. See [24] for more detail on this approach. Each match is scored and if the score is below a given threshold the frame is proposed as a possible loop closure. The loop proposal is then verified with a geometric consistency check, by registering the two frames, using the method described above.

e) Map Estimation: The result of both the visual odometry and visual registration algorithms are inserted into the SLAM map estimation algorithm which is based on iSAM [14].

This approach effectively solves the non-linear least squares (NLLS) problem induced by the pose graph, albeit in an efficient incremental manner. An update is carried out each time new information is added to the graph: either a new node or a constraint between existing nodes. By utilizing the reduced pose graph approach, the rate at which this NLLS problem grows is much reduced, as demonstrated in Fig. 4.

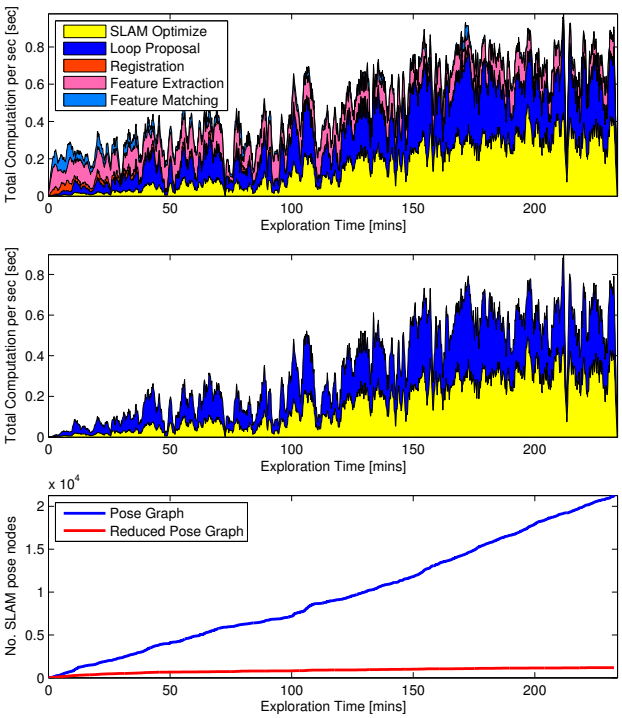
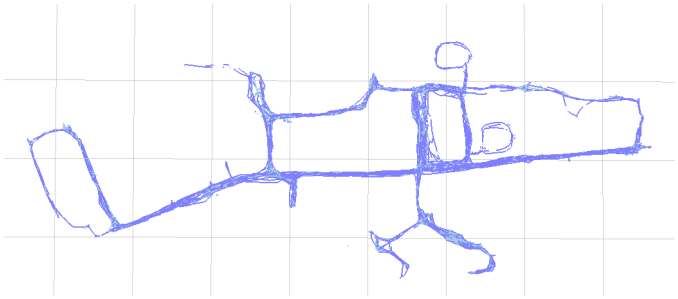
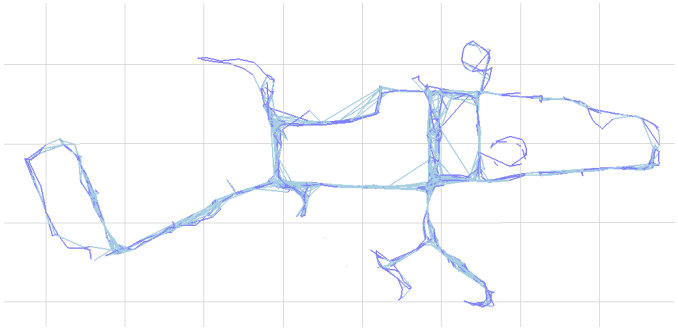


Fig. 4. Timing results when using a pose graph without reduction. The top plot shows the time of each component of the SLAM system as a function of exploration time. The middle plot shows the time for those components that have growing time complexity. The bottom plot shows the number of nodes in the graph – both for the pose graph and the reduced pose graph.



(a) Pose graph – 21000 poses



(b) Reduced pose graph – 1200 poses

Fig. 5. A comparison of a full pose graph vs. a reduced pose graph from 4 hours of traversal. The blue edges are sequential constraints and the light blue are loop closures.

A. Vertical Motion: Elevators

Many buildings contain multiple floors and the robot might be expected to go from one floor to another by elevators. This type of motion is not observable by the vision system or the wheel odometry. Also it is not sufficient to rely on intent only, because the robot does not have full control on which floors the elevator will stop. One could imagine using the loop proposal mechanism, but that might require the robot to exit the elevator.

One possibility is to use a barometer to track vertical motion. Instead, here we use an accelerometer sensor that is present in most robots, often as part of an IMU. Integrating the vertical accelerometer information over time results in the vertical displacement. The method is fairly accurate because the velocity at the start and end of the elevator transit are known to be zero. Results from our floor tracker are show in Fig. 7.

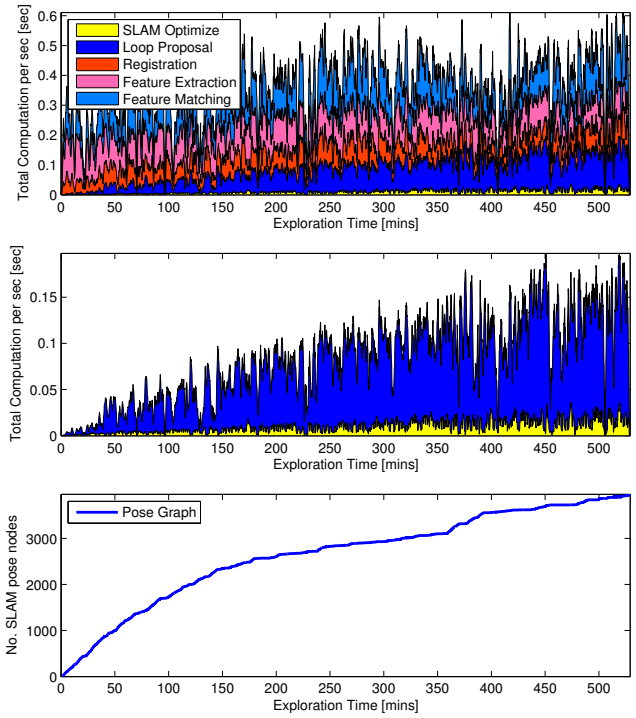


Fig. 6. Timing results when using a reduced pose graph for a 9 hour sequence. The top plot shows the time of each component of the SLAM system as a function of exploration time. The middle plot shows the time for those components that have growing time complexity. The bottom plot shows the number of nodes in the graph.

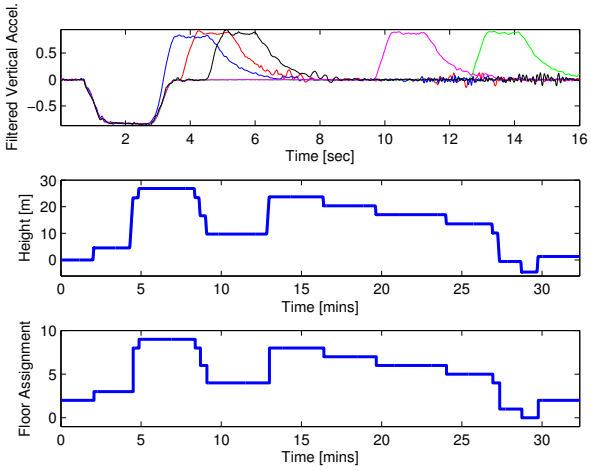


Fig. 7. Top: Using the Z-component of the PR2's accelerometer, the start and end of elevator transitions can be detected using a matched filter. This figure illustrates 5 different elevator rides - showing that the acceleration is clearly repeated. Middle: By integrating this signal the elevation of each floor can be estimated. Bottom: During our 10 floor experiment (beginning and ending on Floor 3), the floor assignment can be determined using a simple lookup table. See Section IV-A for more details.

To assign these vertical displacements to floor numbers, a table of floor heights is maintained. Each time a transition is detected the table is searched for the closest floor. If the distance to that floor is within a given threshold it is accepted as the current floor, otherwise a new floor is added to the table.

Knowing the floor the robot is on is useful for limiting loop closure search. Only matching to nodes on a single floor avoids self-similarities between floors resulting in wrong loop closures.

V. Results

We evaluated the system with vision data that was collected by driving a PR2 through the building. The PR2 was equipped with a stereo camera, a Kinect sensor and a Microstrain IMU among other sensors. The data was collected in a large building over a period of six months. There were repeated excursions through one of the floors (see Fig. 9(b)) and occasional visits to the other floors. In total 10 floors were visited, a map spanning all the floors is shown in Fig. 8.

The robot repeatedly covered the same area and as shown in Fig. 6 the rate at which nodes are added to the graph reduces as time progresses. There are occasional jumps, which are due to new areas being explored. The loop proposal and the optimization grow in complexity as more nodes are added to the graph. Though as shown in Fig. 6 these modules account for only a small fraction of the total running time of the system. The majority of the time is spent on frame registration and feature extraction, both of which are constant time. Visual odometry runs at 30Hz on a separate thread, and loop closure runs at 2Hz.

The accuracy of the mapping system can be seen by comparing the floor plan in Fig. 9(a) with the maps in Fig. 9(b) and Fig. 9(c) that were created using a RGB-D camera and a stereo camera, respectively.

To compare the full pose graph with the reduced pose graph we used a 4 hour dataset. As shown in Fig. 4 when the full pose graph is used, majority of the computation time is spent on optimizing the graph. This will eventually affect the localization accuracy of the system. The graphs created by the two methods are shown in Fig. 5 and they both represent the environment accurately.

One of the lessons learned from this experiment was that it is essential to incorporate more then one sensor input. In our case we used wheel odometry and the IMU. The visual odometry typically failed in the elevators, going through areas were the lights had been turned off, turning around corners when looking at featureless walls, people moving around, etc. Another thing we observed was that by continually incorporating information into the estimate the robot was able to correct the map in later passes if failures had occurred during a previous visit. The conclusion is that these two aspects are important to robust operations.

VI. Conclusion

This paper proposes a reduced pose graph formulation which enables large scale mapping over long time durations. In an environment which has been previously explored, this

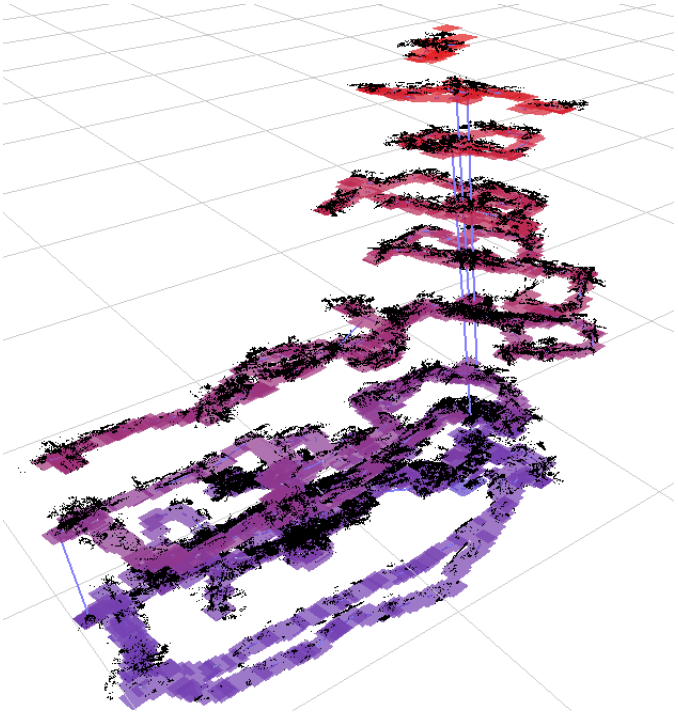


Fig. 8. This figure shows a map of ten floors created from data collected in 14 sessions spanning a four month period. The total operation time was nine hours and the distance traveled was 11km. Elevator transitions are shown as vertical blue lines.

TABLE I
APPROXIMATE FIGURES OF INTEREST CORRESPONDING TO THE 10 FLOOR EXPERIMENT ILLUSTRATED IN FIG. 8.

Duration of Experiment	9 hours
Distance Traveled	11 km
VO keyframes	630K
Failed VO frames	87K
Registrations	303K
Loop proposals	30K

(a) Floor plan of 2nd floor.

(b) Map created with an RGB-D camera.

(c) Map created with a stereo camera.

Fig. 9. Top - A floor plan for one of the floors of the building. Middle - The map constructed using the visual SLAM algorithm using a Kinect camera and projecting the colored pointcloud associated with each pose. Bottom - The map constructed using the stereo vision system. The points are reprojected keypoints in each pose's frame. Each map is approximately 90m across.

approach adds extra pose-to-pose constraints instead of adding new and redundant poses of the underlying pose graph. This important modification allows the SLAM system to scale in size with area of exploration instead of the time of exploration.

The algorithm was demonstrated within a visual SLAM system and its effectiveness was demonstrated on a large dataset where the same environment was frequently re-visited by an exploring robot.

In addition to stereo-vision and RGB-D, information from sensors such as a robot's wheel odometry and IMU can also be incorporated and result in improved robustness in situations where the vision-only system would fail. We have also shown how elevator transits can be detected, enabling seamless multi-floor mapping.

There are still several issues that remain to be explored within the proposed model. Our current implementation cannot fully guarantee that the graph will not grow with time. When the robot becomes temporarily disconnected from the map, it will add poses to graph until re-localized. Changes in the environment can also result in new nodes being added. We believe that this issue can be handled by introspection of the overlapping poses when re-localization finally occurs.

Additionally in future work we aim to evaluate open loop performance when running on a robot in real-time, where the localization and mapping accuracy will directly affect the robot's performance.

REFERENCES

[1] P. Biber and T. Duckett. Experimental analysis of sample-based maps for long-term SLAM. *Intl. J. of Robotics Research*, 28(1):20–33, 2009.

[2] M. Bosse, P. Newman, J. Leonard, and S. Teller. Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework. *Intl. J. of Robotics Research*, 23(12):1113–1139, December 2004.

[3] C. Cadena, J. McDonald, J. Leonard, and J. Neira. Place recognition using near and far visual information. In *Proceedings of the 18th IFAC World Congress*, August 2011.

[4] R.O. Castle, G. Klein, and D.W. Murray. Wide-area augmented reality using camera tracking and mapping in multiple regions. *Computer Vision*

and Image Understanding, 115(6):854 – 867, 2011. ISSN 1077-3142. doi: 10.1016/j.cviu.2011.02.007. URL http://www.sciencedirect.com/science/article/pii/S1077314211000701.

[5] F. Dellaert. Square Root SAM: Simultaneous location and mapping via square root information smoothing. In *Robotics: Science and Systems (RSS)*, Cambridge, MA, 2005.

[6] E. Eade, P. Fong, and M.E. Munich. Monocular graph SLAM with complexity reduction. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3017–3024, October 2010. doi: 10.1109/IROS.2010.5649205.

[7] C. Estrada, J. Neira, and J.D. Tardós. Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Trans. Robotics*, 21(4):588–596, 2005.

[8] J. Folkesson and H.I. Christensen. Graphical SLAM - a self-correcting map. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 383–390, 2004.

[9] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Robotics: Science and Systems (RSS)*, June 2007.

[10] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg. Hierarchical optimization on manifolds for online 2D and 3D mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Anchorage, Alaska, May 2010.

[11] J. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. of the 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, volume 1, pages 318–325, 1999.

[12] A.S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual odometry and mapping for autonomous flight using an RGB-D camera. In *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, Flagstaff, USA, August 2011.

[13] V. Ila, J.M. Porta, and J. Andrade-Cetto. Information-based compact pose SLAM. *Robotics, IEEE Transactions on*, 26(1):78–93, February 2010. ISSN 1552-3098. doi: 10.1109/TRO.2009.2034435.

[14] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. Robotics*, 24(6):1365–1378, December 2008.

[15] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, pages 225–234, Nara, Japan, November 2007.

[16] K. Konolige and J. Bowman. Towards lifelong visual maps. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1156–1163, October 2009.

[17] H. Kretzschmar, C. Stachniss, and G. Grisetti. Efficient information-theoretic graph pruning for graph-based SLAM with laser range finders. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, September 2011.

[18] F. Lu and E. Milios. Globally consistent range scan alignment for environmental mapping. *Autonomous Robots*, 4:333–349, April 1997.

[19] J. McDonald, M. Kaess, C. Cadena, J. Neira, and J.J. Leonard. 6-DOF multi-session visual SLAM using anchor nodes. In *European Conference on Mobile Robotics*, Örbero, Sweden, September 2011.

[20] M.J. Milford and G.F. Wyeth. Persistent navigation and mapping using a biologically inspired SLAM system. *Intl. J. of Robotics Research*, 29(9):1131–1153, August 2010.

[21] P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schroter, L. Murphy, W. Churchill, D. Cole, and I. Reid. Navigating, recognising and describing urban spaces with vision and laser. *Intl. J. of Robotics Research*, 28, October 2009. doi: 10.1177/0278364909341483.

[22] K. Ni, D. Steedly, and F. Dellaert. Tectonic SAM: Exact, out-of-core, submap-based SLAM. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1678–1685, April 2007.

[23] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2262–2269, May 2006.

[24] P. Pinies, L.M. Paz, D. Galvez-Lopez, and J.D. Tardos. CI-Graph SLAM for 3D reconstruction of large and complex environments using a multicamera system. *J. of Field Robotics*, 27(5):561–586, Sep/Oct 2010.

[25] K. Pirker, M. Ruether, and H. Bischof. CD SLAM - continuous localization and mapping in a dynamic world. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, September 2011.

[26] G. Sibley, C. Mei, I. Reid, and P. Newman. Planes, trains and automobiles – autonomy for the modern robot. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 285–292. IEEE, 2010.

[27] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.

[28] M.R. Walter, R.M. Eustice, and J.J. Leonard. Exactly sparse extended information filters for feature-based SLAM. *Intl. J. of Robotics Research*, 26(4):335–359, 2007.

[29] Z. Wang, S. Huang, and G. Dissanayake. D-SLAM: A decoupled solution to simultaneous localization and mapping. *Intl. J. of Robotics Research*, 26(2):187–204, 2007.